Internet Resource Discovery Services

Katia Obraczka, Peter B. Danzig, and Shih-Hao Li
University of Southern California

What do you do when you want to find data in an overwhelming sea of information? The authors help by describing Internet resource discovery services.

A few years ago, researchers started new projects by requesting bibliographic searches, contacting fellow researchers, and leafing through conference proceedings and university technical report lists. Today, the Internet can serve all these functions and serve them well, but trying to locate data in this sea of information can be a considerable task.

Resource discovery services now help users locate and retrieve information. These services contain tools for browsing, searching, and organizing information distributed throughout the Internet. Browsing tools let users navigate the information space to find the specific data they need. Indexing search tools automatically locate relevant data on the basis of user interest. Independent of the approach used, resource discovery services can also help users organize newfound information so that they can refer to it without having to repeat the entire discovery process.

We present an overview of the resource discovery services currently available on the Internet. Because resource discovery has been the subject of intense research, this article is not meant to be exhaustive.

The Wide Area Information Servers project

Dow Jones and Company, Thinking Machines Corp., Apple Computer, and KPMG Peat Marwick developed the WAIS project[1] because they were interested in the information discovery-and-retrieval problem. The project seeks to provide users with a uniform, easy-to-use, location-transparent mechanism to access information.

WAIS is a full-text information-retrieval architecture whose clients and servers communicate through an extension of the Z39.50 protocol standard from the National Information Standards Organization.

WAIS architecture. Figure 1 is a schematic view of the architecture. WAIS Clients translate user queries into the WAIS protocol and query the Directory of Services for relevant databases. WAIS then transmits the request to a selected set of databases over the communications network. Database servers keep complete inverted indexes on stored document contents and execute full-text searches on them. In response to a query, a server returns a list of relevant object descriptors.

0018-9162/93/0900-0008$03.00 © 1993 IEEE

These descriptors correspond to documents that contain words specified in the user query. The client displays query results and can retrieve documents from corresponding servers. Clients also display a numerical score for each hit. The score relates to the frequency of query-specified words in the object's contents and provides feedback to help users refine their queries.

WAIS databases. One server indexes the descriptors of the other servers. This yellow page (as in the telephone book) service resides at a well-known address, and users query it in the same way as they do other servers. Instead of referencing documents, however, it references participating databases. When a new information provider wants to join WAIS, it must submit its location, description, and other relevant information to the directory server. The WAIS Directory of Services currently contains around 300 registered databases. The WAIS directory server's database is replicated on a number of servers and has its primary copy stored on host think.com.

Discovery session. Figure 2 illustrates a WAIS discovery session running on an X Windows client. The query first arrives at the Directory of Services (see the "In Sources" box in the upper window), which responds with descriptors of pertinent servers. The user can then select servers from this set and submit the query to them (see the "Resulting documents" area in the upper window). In this example, two servers are selected and queried (see the "In Sources" box in the lower window). They return a set of document descriptors from which the user can identify relevant documents to be retrieved. To refine the search, users can select one or more of the returned documents and place them in the "Similar to" box. When the query is rerun, WAIS updates the results to include similar documents. It ranks similarity in terms of how many words match.

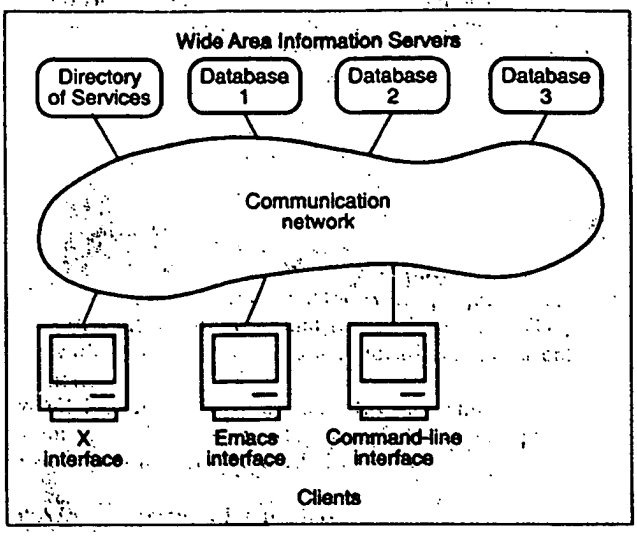

Figure 1. The WAIS architecture.

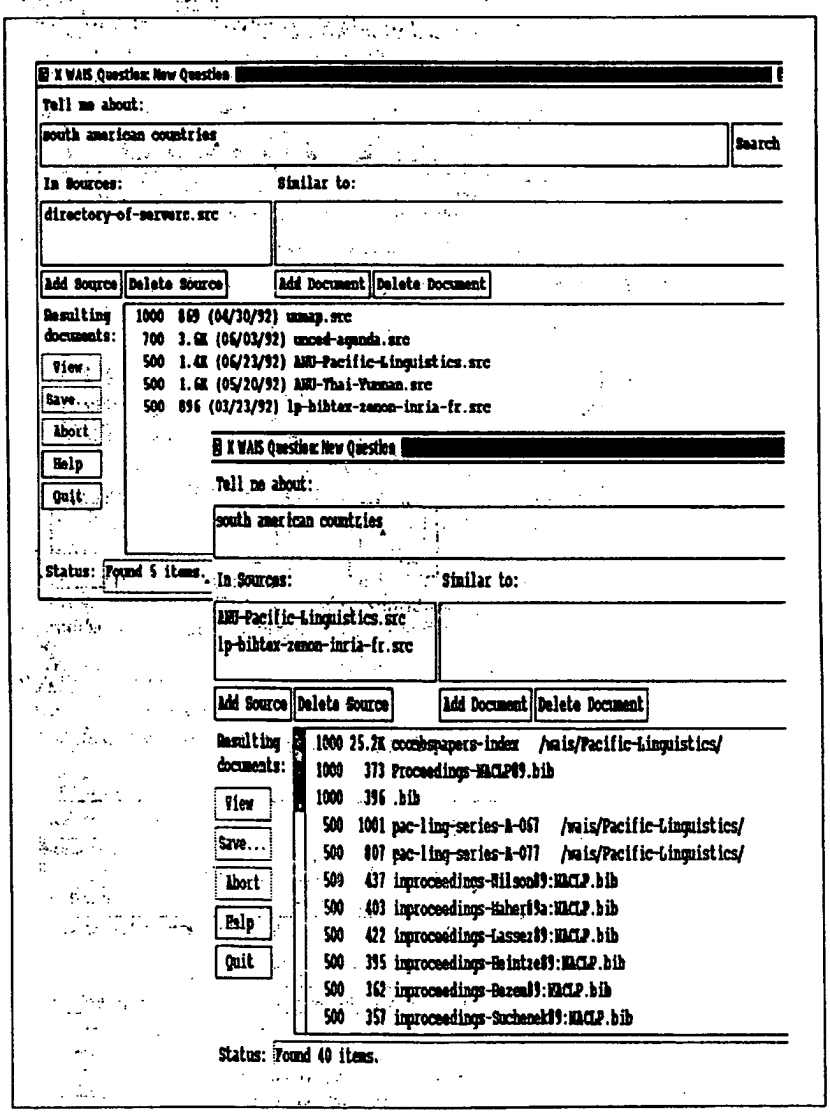

Figure 2. An example WAIS discovery session.

Archie

The Archie[2] service addresses the problem of locating files by attribute (files are currently listed only by their names) in the Internet. Archie was developed at McGill University by Alan Emtage and Peter Deutsch . Archie servers centralize

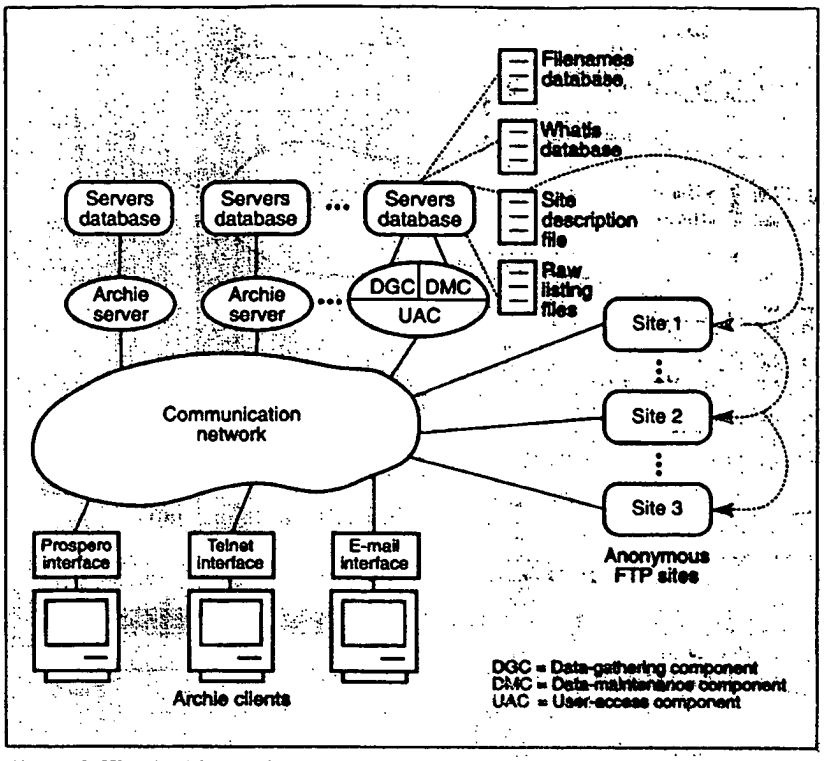

Figure 3. The Archie architecture.

indexing information on file-name data distributed throughout thousands of Internet public archive sites.

Archie databases. Archie currently offers two databases: Filenames and Whatis. The Filenames database indexes the names of files available from hundreds of Internet file-transfer protocol (FTP) sites. FTP lets users retrieve files stored on Internet hosts. Anonymous FTP, which does not require the user to have an account on the FTP site, is widely used for distributing free software and documents. Archie automatically updates entries in the Filenames database. Users can query this database for file names that match

- specified patterns,
- a list of FTP archive sites, or
- a list of the files available from specific sites.

The Whatis database contains the names and descriptions of software packages, documents, and other information available on the Internet. Entries include text strings consisting of keywords and associated descriptions. Users can perform case-insensitive text-string searches that are applied to both keywords and corresponding descriptions.

The Whatis database is manually maintained by the system/database administrator. Information is gathered from secondary sources (such as Usenet postings and an author's e-mail submission) and entered into the database by hand. In contrast, data-gathering and data-maintenance components maintain the Filenames database.

Archie architecture. Archie clients access both databases through the user-access component (UAC). The data-gathering component (DGC) relies on FTP site administrators to find out about new FTP archives. Every time a new FTP archive is reported, an entry corresponding to the new site is added to the site descriptions file, which lists all known FTP sites. Periodically, the DGC connects to each known FTP site and fetches a recursive listing of its contents. This information remains on the Archie server in the raw listing files until it is processed by the data-maintenance component (DMC), which converts the listings into a format that can be added to the Filenames database. Figure 3 presents the Archie architecture.

The UAC lets Internet users access and query Archie servers. Currently, access is possible via electronic mail, Telnet, and Prospero.[3] E-mail users can submit a query by sending a message to an Archie server, which sends a message back to the user with the query results. Telnet users connecting to an Archie server through the telnet command can submit queries to both databases. Each telnet session requires a significant amount of server resources. For this reason, Archie currently uses Prospero as a front end for each Archie server. The Prospero server lets users access Archie databases without logging directly on to the Archie server. The Prospero interface also lets Archie clients use the Prospero protocol. One such client is Xarchie, which provides a point-and-click interface to Archie. In the commercial version Archie 3.0, all available interfaces are clients of the Prospero interface, which communicates with Archie servers that have Prospero front ends.

Archie session. Figure 4 shows an Archie session using the Xarchie client. In the upper window, the user specifies a pattern to be matched. Archie then returns a list of archival sites whose names match the pattern. As illustrated by the bottom window, when the user selects a specific site, the file's complete path name, size, access permissions, and date of last modification are also displayed (see the lower portion of the bottom window). Users can then request that the file be "FTP'ed" to their local site.

Traffic. Archie consists of a dozen servers around the Internet. Originally, Archie servers kept their databases consistent by copying the site-listing information from the main server in Montreal, Canada. According to Emtage and Deutsch,[3] the update mechanism and user queries to the main Archie server generated approximately 50 percent of all Montreal-bound Internet traffic. Consequently, the task of polling the participating FTP sites is currently distributed among various Archie servers, which execute a flooding-based consistency-maintenance protocol among themselves.

Archie's developers have described it as a low-tech solution to the resource discovery and information-retrieval problem. Archie's simplicity and use of existing mechanisms have been key to

its success. After approximately two years in service, with over 1,000 registered FTP archive sites that offer some 2,100,000 files and 3,500 software packages, Archie is accessed from 47 countries about 50,000 times per day.

Prospero

The Prospero File System,[3] a tool for organizing distributed information, was originally developed at the University of Washington by Clifford Neuman. It lets users build customized "views," or virtual systems, of directories located throughout the Internet.

Organization. The Prospero name space forms a generalized directed graph, in which intermediate nodes are directories, leaves are files, and edges are Prospero links. Just like traditional distributed file systems, subtrees of the Prospero name space can be stored on different Prospero servers. A user's name space corresponds to the subgraph starting at a particular node, which serves as the root of the user's name space.

Users organize their name space by building views, which are essentially directories composed from various sources that include the user's own views and imported views from other users. These directories can reside on different Prospero servers.

In Prospero, an index is a special type of view that returns a directory of objects that satisfy some query. It allows Prospero users to access other search engines. For example, the Prospero-Archie interface lets users build views containing directories with objects resulting from Archie queries.

Users find information by navigating through available views. The Prospero client provides users with navigational tools analogous to the ones provided by traditional file systems. One command allows users to change the current virtual directory to the one specified in the command. Another command displays the name of the current virtual directory and describes its physical location. Users can also list the contents of a virtual directory.

Prospero session. Figure 5 on the next page shows a sample session that illustrates the Prospero-Archie interface. Starting at the root of the Prospero name space, the user can then change the current virtual directory, display its name, and list its contents by using the cd, pwd, and ls commands. To submit an Archie request to find all file names that match the string "wais," the user changes the directory to /Archie/wais and lists the results by using the ls command.

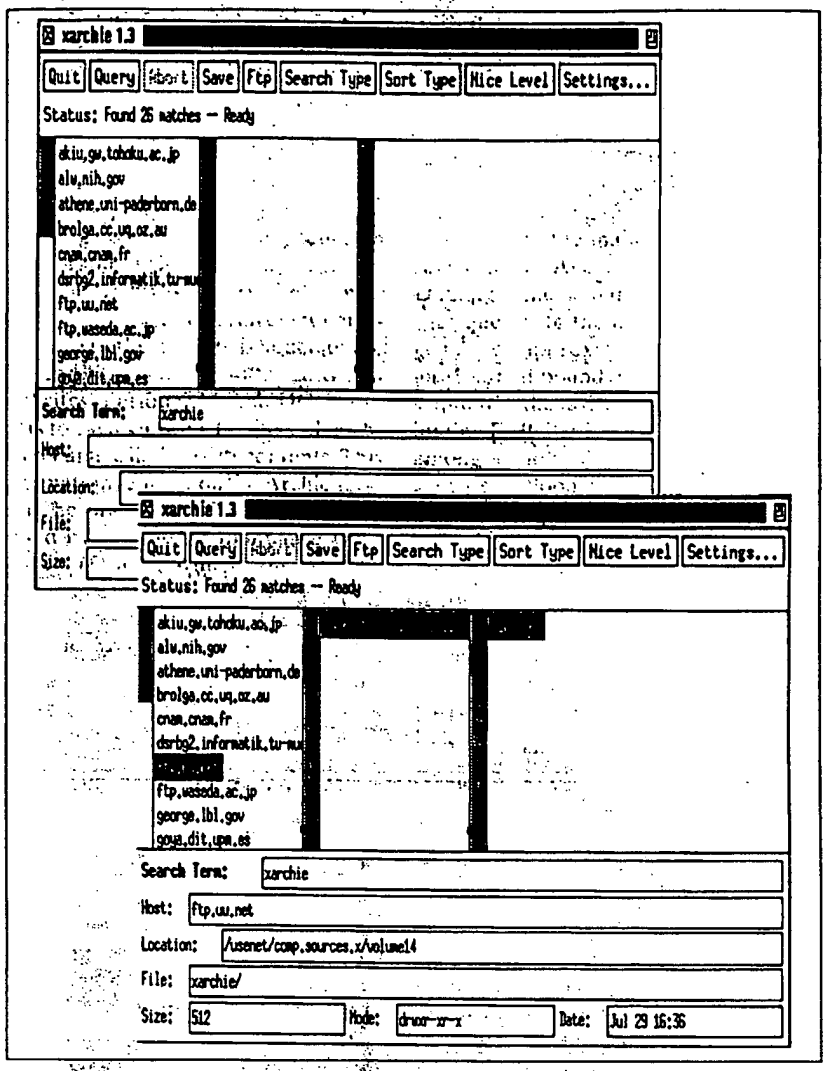

Figure 4. An example Archie session using Xarchie.

Links. When users find an interesting object, they can include it in their view by linking to it. The Prospero client provides commands to add and delete links from the current node in the user's name space to or from a target node or leaf. A Prospero link specifies the name of the host where the object is stored and the local name of the object on that host. If the link's target is a directory, the link provides information to resolve a name in that directory by querying the corresponding server. For files, associated links contact the appropriate server to provide access-mode information. Prospero supports Sun's Network File System, the Andrew File System, anonymous FTP, Gopher, and WAIS.

A link also provides information such as its type and associated filters. Special links allow the contents of the target directory to be included in the directory containing the link. By associating filters with links, users can build customized views from existing ones. A filter customizes the target view by reorganizing or extracting parts of it. Listing a Prospero view requires a computation distributed across all nodes reachable by transitive closure of all the view's links, indexes, and filters.

```
% cd /

% ls
afs                 info              projects
Archie              lib               releases
databases           mailing-lists     sites
documents           newsgroups
guest               papers

% cd papers

% ls
authors             conferences       subjects
bibliographies      journals          technical-reports

% cd journals

% ls
acm-sigcomm-ccrieee-tcos-nl

% cd acm-sigcomm-ccr

% ls
application.ps      jan89             jul90          sigcomm90-reg.ps
apr89               jan90             oct88
apr90               jan91             oct89
apr91               jul89             sigcomm90-prog.ps

% cd /databases

% ls
bibliographies      faces             machine-learning

% cd /Archie/wais

% ls
wais                wais-8-b3-ses.tar.Z  wais-8-b3-ses.tar.Z
wais-8-b3.README    wais-8-b3.tar.Z   wais-minutes-92mar.t
wais.library        wais.src          wais.txt
wwais102.zip
```

Figure 5. A sample Prospero session.

Virtual system registration. Users "advertise" information by registering their virtual systems with the Prospero server administrator. The administrator creates a link to the new virtual system in the master view of virtual systems so that other users can see, navigate, and link to accessible portions of it.

Usage. Currently, there are close to 50 Prospero servers. Most users (from more than 10,000 systems in 30 different countries) run as Prospero clients.

Gopher

Gopher,[4] developed at the University of Minnesota, lets users search and browse distributed information on the Internet. Gopher organizes information into a hierarchy in which intermediate nodes are directories, or indexes, and leaf nodes are documents. (Actually, the Gopher information space is a directed graph, since it allows cycles.) Users navigate the Gopher information space guided by the Gopher client's hierarchical menu system.

Gopher architecture. The architecture, as shown in Figure 6, consists of clients and servers communicating through the Gopher protocol, which is implemented on top of TCP-IP (transmission-control protocol-Internet protocol).

The root of Gopher's hierarchy resides on host rawBits.micro.umn.edu at the University of Minnesota. This is the default directory retrieved by Gopher clients when they are first invoked. Clients can also be configured with other entry points into the hierarchy. The Gopher root server has knowledge of all top-level services and advertises their existence to users. The architecture contains essentially one top-level server per participating organization, such as university campuses, private corporate institutions, or government agencies. Lower level servers can be linked to the corresponding top-level server, so that once users find the appropriate top-level server, they can navigate through the hierarchy by following the links to the lower level servers.

For example, university campuses running Gopher servers may register a central top-level server with the root server. Each university's central Gopher server can link to existing departmental servers, which in turn can link to lower level servers.

Gopher objects are identified by type, user-visible name, server's host name and port number, and the object's absolute path name within the server file system. The user selects an object on the basis of its user-visible name, and the client retrieves it by constructing a "handle" from the server's host name, its port number, and the object's path name. Users can then navigate through the available information space that contains full-text document objects, which are stored as

- files in the corresponding servers, and
- directory objects that can be distributed across multiple servers.

Full-text search operations can also be performed. Gopher's search servers maintain full-text inverted indexes of subsets of the documents stored in a Gopher server. Search servers can be configured to index more than one server. For instance, an Internet request for comments (RFC) full-text search server indexes all existing RFCs and executes keyword searches on their contents. A full-text search server returns to the client handles to documents that match a Boolean search pattern. Gopher clients can also retrieve objects from WAIS, Archie, and FTP servers.

Gopher session. Figure 7 illustrates a

sample Gopher session. Starting at the root directory, the user traverses the Gopher information space by selecting interesting directories, such as Libraries/ and Library of Congress Records/, or executing full-text searches on indexes like "search Library of Congress records from 12-91 to present."

World-Wide Web

The World-Wide Web,[5] developed at CERN in Switzerland, merges information discovery and hypertext techniques. It organizes data into a distributed hypertext in which nodes are either full-text objects, directory objects called cover pages, or indexes. WWW also supports full-text searches over documents stored at a particular server.

WWW architecture. The architecture is based on the client-server model, with WWW clients providing users with a hypertext-like browsing interface. Besides its native hypertext transfer protocol (HTTP), WWW clients understand FTP and the network news transfer protocol (NNTP). FTP lets users access file archives on the Internet, where file directories are browsed as hypertext objects. NNTP allows access to Internet news groups and news articles. News articles often contain references to other articles or news groups, which are represented as hypertext links.

HTTP allows document retrieval and full-text search operations. It runs on top of TCP and maps each request to a TCP connection. HTTP objects are identified by the HTTP protocol type, the corresponding server's name, and the path name to the file where the objects' contents reside. Parts of documents can also be specified. If a search operation is requested, the HTTP object identifier carries the set of specified keywords instead of a path name. Future implementations of the HTTP protocol will include data-format negotiation between client and server. Currently, only plain text and simple hypertext formats (Hypertext Mark-up Language, or HTML) are implemented.

Discovery trees. Information accessible through WWW can be seen through three discovery trees. The first one classifies information by subject. An entry in the WWW root directory links the directory to the current subject-classification tree. This classification includes topics such as aeronautics, astronomy, biological sciences, computer sciences, and humanities. This information is spread across all kinds of servers, including WAIS, Gopher, and WWW. Because WWW was created for the high-energy physics community, a special entry in the root directory links to a cover page that contains the existing HTTP servers specializing in the subject. As they become available, indexes to other disciplines are added by the root directory's database administrator.

The second WWW discovery tree classifies by server type. The cover page corresponding to this classification lists all servers available through WWW. This includes entries for WAIS, Gopher, NNTP, and WWW servers. There is even an entry for anonymous FTP sites that can be searched through Archie. The third tree, which classifies by organization, is not very populated in the sense that it does not contain a great deal of information.

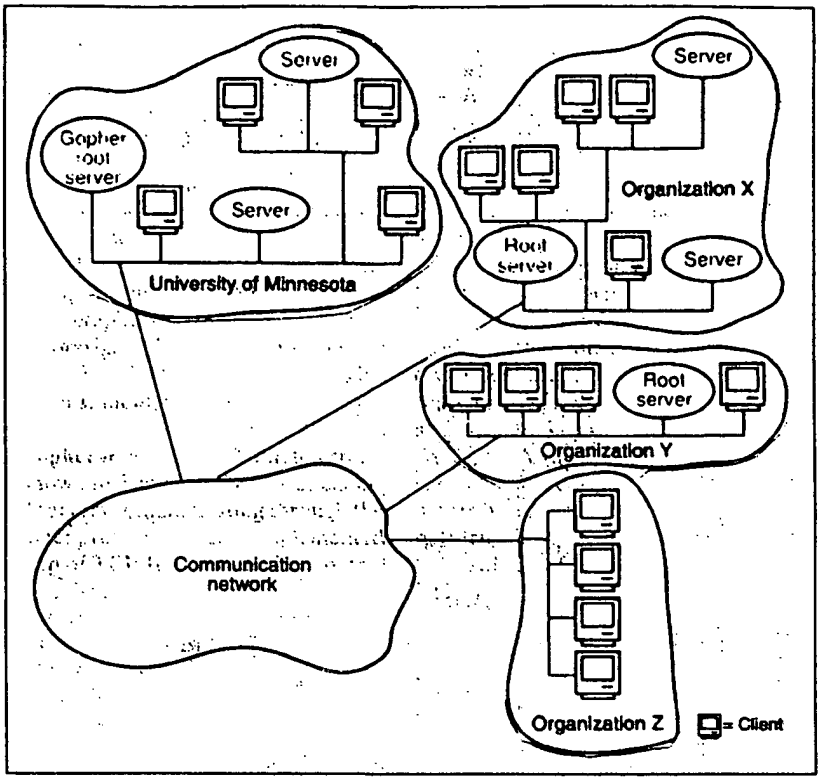

Figure 6. The Internet Gopher architecture.

Personalized web. The WWW discovery trees correspond to the different ways information is organized and can be discovered. Discovery sessions involve users starting on their home cover page, following a link to an index, executing a search, and following the resulting links. As users find interesting information, they build their personalized web by linking to nodes in the global web. Currently, the default cover page, which resides on host info.cern.ch and represents the root of the WWW information space, is the one retrieved by the WWW client when it is invoked. However, users can customize their home cover page so that they can start anywhere in the WWW information space. Figure 8 presents an example session.

Publishing information. Making information available through WWW can involve a similar discovery task. The information publisher tries to find the appropriate cover page to reference the new data. Then the publisher contacts the person responsible for that cover page, who adds a link to the new data. The publisher can also run a new server, which requires the new server's administrator to contact WWW administrators to add the new server to the list.

Servers and hosts. Currently, besides WAIS and Gopher servers, there are about 24 WWW servers accessible to WWW clients. The WWW server on info.cern.ch has logged access from approximately 6,000 hosts that use their own WWW clients or connect to the publicly available client.

Resource discovery at the University of Colorado

The Resource Discovery project at the University of Colorado-Boulder is best known for Netfind[6] but has also investigated various issues in the resource discovery arena.

Netfind. This white pages directory tool tries to locate information about an Internet user when given the user's name and organization. A successful Netfind query, like the one shown in Figure 9, returns information such as the user's e-mail address and telephone number.

Netfind builds its indexing database, called the seed database, by using data scattered across multiple existing sources, such as network news messages, the domain naming system (DNS), the simple mail-transfer protocol (SMTP), and the finger utility. The seed database keeps organization names, city names, and corresponding host names gathered from news message headers. On the basis of organization names and city names provided to Netfind, matching host names are selected from the seed database. DNS is then used to locate authoritative name servers for the domains to which the selected hosts belong. Each name server located is queried by using SMTP to find mail-forwarding information about the specified user. If found, the corresponding hosts are probed by using the Unix finger utility. To improve Netfind's response time and increase its resiliency to host and network failures, up to 10 threads can be used to allow sets of DNS, SMTP, and finger-query sequences to be executed in parallel.

Other projects. Other resource discovery-related projects under development at the University of Colorado include a network visualization tool that focuses on discovering information

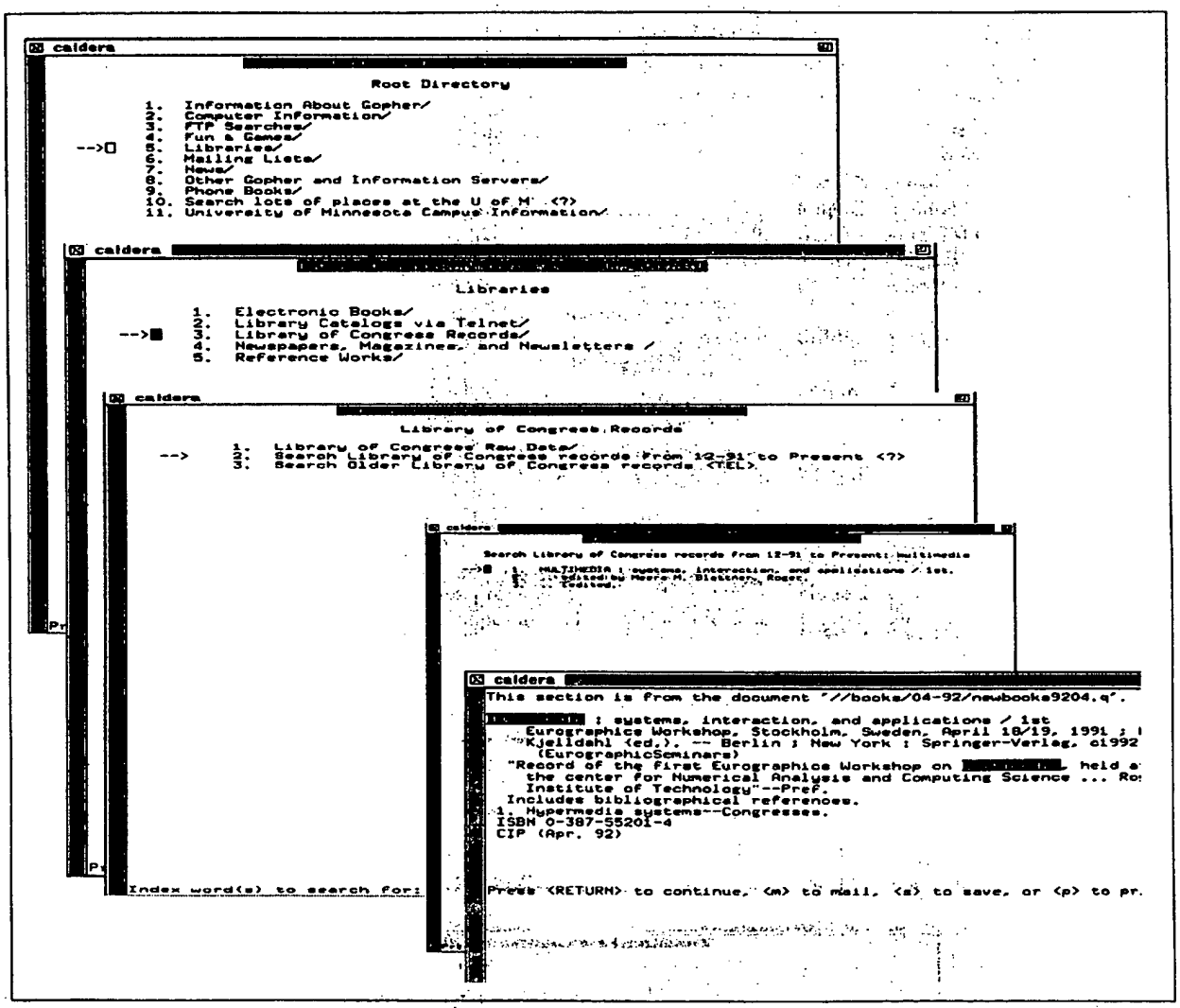

Figure 7. A sample Gopher session.

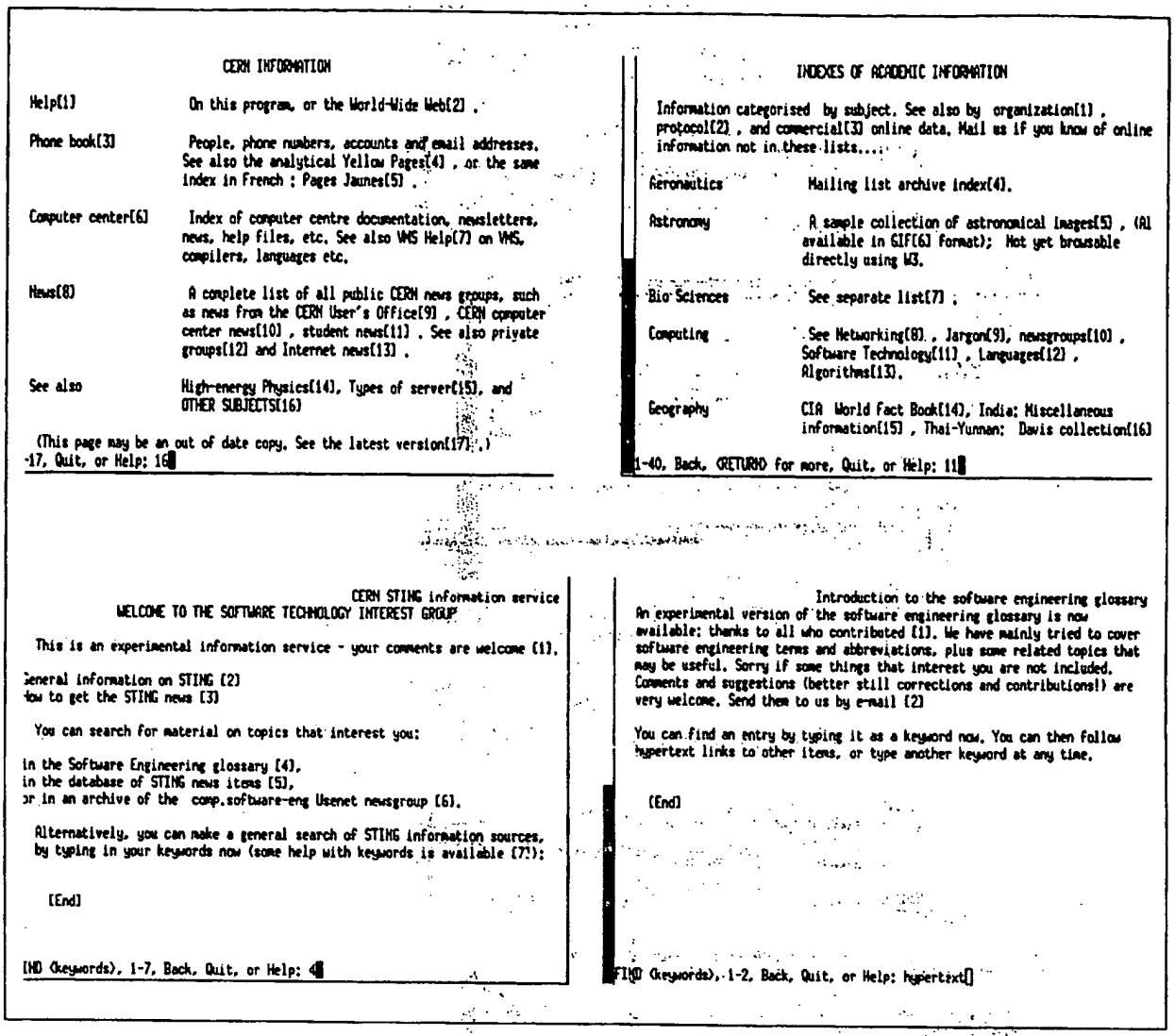

Figure 8. A sample WWW session.

about networks. Items include topology, congestion, routing, and protocol usage. A global e-mail study focuses on providing ways of locating users with particular interests or expertise on the basis of e-mail pattern analysis.

X.500

The X.500 directory[7] resulted from standardization efforts in the field of directory services by the CCITT (International Consultative Committee for Telephony and Telegraphy) and the ISO (International Standards Organization).

Attribute queries. Unlike the domain-naming system, which basically maps host names into corresponding Internet addresses and vice versa, X.500 entries consist of a set of attribute-value pairs. X.500 accepts attribute-based queries. The directory's name space is hierarchically organized and distributed among its servers. Administrative authority over portions of the global name space is delegated to different autonomous organizations, which can transfer authority over portions of their assigned subtrees. As in DNS, portions of X.500's name space are replicated on different servers, which use a simple replication mechanism based on designated slave and master servers. Figure 10 illustrates a sample X.500 session.

Internet offers numerous freely distributable X.500 software packages. Some of the X Window System interfaces are Bellcore's Xdi, Csiro's xdua, and the University of Wisconsin's xwp. An X.500-specific WAIS database called x.500.working-group.src provides more information on X.500 software and documentation.

Indie

Distributed Indexing, or Indie for short,[8] is a distributed information discovery-and-retrieval architecture we developed at the University of Southern California. Indie consists of a replicated directory of services and a collection of broker databases. Brokers automatically cluster references to re-

lated information by indexing their own data, as well as data stored in other brokers, databases, and other discovery tools. As in Archie, this clustering of indexing information lets users efficiently search all participating databases. In addition, because it was built on top of the Distributed Hypertext (DHT) system data model and communication protocol, Indie inherits the organizational capabilities of hypertext systems.

Object descriptors. An Indie broker stores descriptors of objects relevant to the topic in which it specializes. It extracts them from other Indie brokers and primary data sources that it indexes. An object descriptor contains an arbitrary number of attribute-value pairs describing the object. Examples of attributes that constitute an object descriptor include bibliographic information about the object such as the name of the author(s), the document title, and publication date.

The object descriptor includes technical data about the object, such as object type, object identifier, and the time stamp assigned to each object by the database that created it. It also includes some attributes used by Indie's consistency maintenance mechanism. The object descriptors need not include the object itself. This lets a service advertise an object while retaining access control.

```
caldera.usc.edu% telnet bruno.cs.colorado.edu

login: netfind

================================================
Welcome to the University of Colorado Netfind server.
================================================

Alternate Netfind servers:
   archie.au (AARNet, Melbourne, Australia)
   bruno.cs.colorado.edu (University of Colorado, Boulder)
   ...
   su.uakom.cs (Slovak Academy of Sciences, Czech and Slovak Fed. Repub.)

Top level choices:
   1. Help
   2. Search
   3. Seed database lookup
   4. Options
   5. Quit (exit server)
—> 2
Enter person and keys (blank to exit) —> obraczka usc

There are too many domains in the list.
Please select at most 3 of the following:
   0. hsc.usc.edu (university of southern california, los angeles)
   1. hsc.usc.ed.ln.net (los nettos, usc information sciences institute, marina del rey, california)
   2. ln.net (los nettos, usc information sciences institute, marina del rey, california)
   3. usc.vcu.edu (virginia commonwealth university, richmond)
   4. usc.edu (university of southern california, los angeles)
   5. usc.co.jp (usc corporation, yokohama, japan)
   6. usc.ed.ln.net (los nettos, usc information sciences institute, marina del rey, california)
   7. usc.e.ln.net (los nettos, usc information sciences institute, marina del rey, california)
   8. usc.es (unspecified)
Enter selection (e.g., 3 1 2) —> 4
( 0) check_name: checking domain usc.edu. Level = 0
MAIL IS FORWARDED TO obraczka@chaph.usc.edu
NOTE: this is a domain mail forwarding arrangement - so mail intended
   for "obraczka" should be sent to "obraczka@usc.edu"
   rather than "obraczka@chaph.usc.edu".

( 0) check_name: checking host chaph.usc.edu. Level = 0
SYSTEM: chaph.usc.edu
   Login name: obraczka      In real life: Katia Obraczka
   Directory: /home/chaph2/obraczka    Shell: /usr/local/etc/notallowed
   Last login Sun Apr 7, 1991 on ttyp1 from jerico
   No unread mail
   No Plan.

( 0) Attempting finger to current indication of most recent "Last login" machine jerico.usc.edu
( 0) check_name: checking host jerico.usc.edu. Level = 1

SUMMARY:
- Among the machines searched, the machine from which user
  "obraczka" logged in most recently was jerico.usc.edu,
  on Sun Apr 7, 1991.
- The most promising email address for "obraczka"
based on the above search is
obraczka@usc.edu.
```

Figure 9. A sample Netfind session.

Generator objects. A generator object describes an Indie broker. The generator consists of a textual abstract and a Boolean expression over a set of bibliographic fields (we call the Boolean expression the generator rule). It also contains fields such as the broker's location, the size of its database, and the object identifier corresponding to the generator object. The name-generator rule expresses the idea that a broker's database is generated and periodically updated by evaluating the rule over a number of other brokers and primary data sources.

To become visible to users and other brokers, all brokers register their generator object with Indie's directory of services. A broker can register itself with any replica of the directory of services. As part of the registration procedure, the selected replica returns a list of other generator objects pertinent to the new broker. The broker stores this list in its registration table and refers to it when choosing the set of databases to index. The directory of services replica also reports changes to the broker's registration table as new brokers join in or participating brokers cease to exist. From time to time, broker administrators use the administrator's tool to refer to the corresponding registration table and decide whether to index new brokers.

Brokers indexed by other brokers store the indexing brokers' generator objects in their trigger table. The name "trigger table" refers to active databases in which specific rules are triggered and evaluated when the database changes in specific ways. When one broker registers its generator object with another broker, the indexed broker executes the generator rule and reliably forwards the retrieved set of object descriptors to the indexing broker. Afterwards, adding or deleting objects from the indexed broker's database can trig-

ger the evaluation of generator rules in its trigger table, causing these changes to be forwarded to the corresponding indexing brokers.

The directory of services is simply a specialized broker. When a broker registers itself with a replica of the directory of services, the broker's generator object is stored in that replica's trigger table. Only the updates to the directory of services that trigger this rule are forwarded to the broker.

Gateways. Non-Indie servers attach to Indie through a gateway broker. Indie provides a gateway library consisting of a set of routines which non-Indie servers can call to communicate with their gateway broker. The gateway broker itself is just a normal Indie broker modified to communicate with non-Indie servers. It manages a trigger table, a registration table, and the interface to the directory of servers. A non-Indie server can efficiently attach itself to Indie by making appropriate calls to the gateway library. Otherwise, the gateway broker must communicate with the non-Indie server in its native protocol and poll for updates by periodically extracting indexing information from the server.

Indie architecture. Figure 11 illustrates an example indexing configuration and its operation. Suppose that a user wants to find all recent technical publications on distributed operating systems. The user interface submits the corresponding user query to a replica of the directory of services, which evaluates the query against its generator object database. This computation produces a list of brokers whose descriptions are pertinent to the user query. In this example, the directory of services could return a reference to the operating systems broker and the distributed systems broker. The user interface ranks the list of target brokers according to interest. This ranking procedure could be based on counting the number of keyword matches in the broker descriptor. In this case, the user interface could send the query to both brokers, who — after evaluating the user query on their indexing information database — return a list of appropriate object descriptors. On the basis of this information, the user can choose to retrieve a copy of one or more interesting articles from one or more full-text retrieval systems.

For instance, a user decides to retrieve a copy of an interesting report from the University of California at Los Angeles. The user interface contacts the corresponding gateway to retrieve the selected object. The user can also permanently link any node in the user information space to the UCLA technical report for future use.

Consistency mechanism. Indie addresses database consistency and recovery with a time-stamped augmented flooding algorithm that also permits convenient recovery from network partition, operating system crashes, and media failure of the broker's database. Indie's time stamp-based consistency mechanism works as follows. All trigger table entries and registration table entries are time-stamped. When an object is added to or deleted from a broker, the change causes the broker to evaluate certain rules stored in its trigger table against updates with time stamps earlier than the trigger table entry. The broker forwards the appropriate changes to each of its affected peers (that is,

```
%dish
Welcome to Dish (DIrectory SHell)

Dish -> squid
Connected to Incan Speckled Iguana at '0101'H/Internet=
128.129.32.31+17003
Current position: @c=US@st=California@o=Information Sciences
Institute
User name: @
Current sequence: default

Dish -> list
1 organizationalUnitName=Business Office
2 organizationalUnitName=HPCC
3 organizationalUnitName=Info Processing Center
4 organizationalUnitName=Integrated Systems
5 organizationalUnitName=Intelligent Systems
6 organizationalUnitName=Silicon Systems
7 organizationalUnitName=Software Sciences
8 commonName=Manager
9 commonName=Postmaster

Dish -> moveto 2

Dish -> search Hotz
objectClass - organizationalPerson & pilotObject & newPilotPerson
& quipuObject
commonName - Steven Hotz
commonName - hotz
surname - Hotz
postalAddress - Information Sciences Institute
Suite 1001
4676 Admiralty Way
Marina del Rey, CA
90292
telephoneNumber - +1 310-822-1511 x402
facsimileTelephoneNumber - +1 310-823-6714
userid - hotz
rfc822Mailbox - hotz@venera.isi.edu
otherMailbox - internet: hotz@venera.isi.edu

Dish -> quit
%
```

Figure 10. A sample X.500 session.

each registered indexing broker corresponding to the triggered generator rules) and advances the trigger table time stamps to the time stamp of the latest update. It then transmits this time stamp to all affected peers, and they record it in the appropriate registration table entry. When the indexed broker cannot establish communication with a peer, it marks the trigger table entry as out of date. Time stamps of rules neither triggered nor out of date are advanced to the current time. Finally, peers occasionally poll one another in an attempt to maintain consistency.

Replication. Indie's indexing mechanism causes "lazily consistent" broker replication as a side effect. To replicate a broker, we create a new broker to serve as the replica, assign the replica the same generator rule as the broker to be replicated, and have the replica index the broker or some of its other replicas. Since the replica shares the same generator rule as the primary copy, it fills with the same data. Indie's update-and-recovery algorithm guarantees that all replicas eventually learn of the update.

Replication in the context of the directory of services considers all replicas equal. This means that there is no primary copy of the directory of servers. Instead, clients register or unregister themselves with the replica of their choice. All replicas participate in a flooding-based consistency maintenance mechanism to keep their databases consistent.

Status. We have completed Indie's first implementation phase. Currently, the prototype consisting of Indie brokers, a centralized directory of services, and gateways to FTP archives runs on the University of Southern California's Network and Distributed Systems Laboratory. We are now implementing Indie's consistency maintenance and replication mechanisms.

Other research initiatives

Other discovery tools include the Knowbot Information Service,[9] Alex,[10] Semantic File Systems,[11] and Nomenclator.[12]

Knowbot Information Service. The Digital Library System (DLS), proposed by the Corporation of National Research Initiatives, is an open architecture whose goal is to integrate access to existing and future information sources on the Internet. Knowbots, an abbreviation for Knowledge Robots, are active components of the DLS. A Knowbot is an intelligent program that can exchange messages with other Knowbots, move

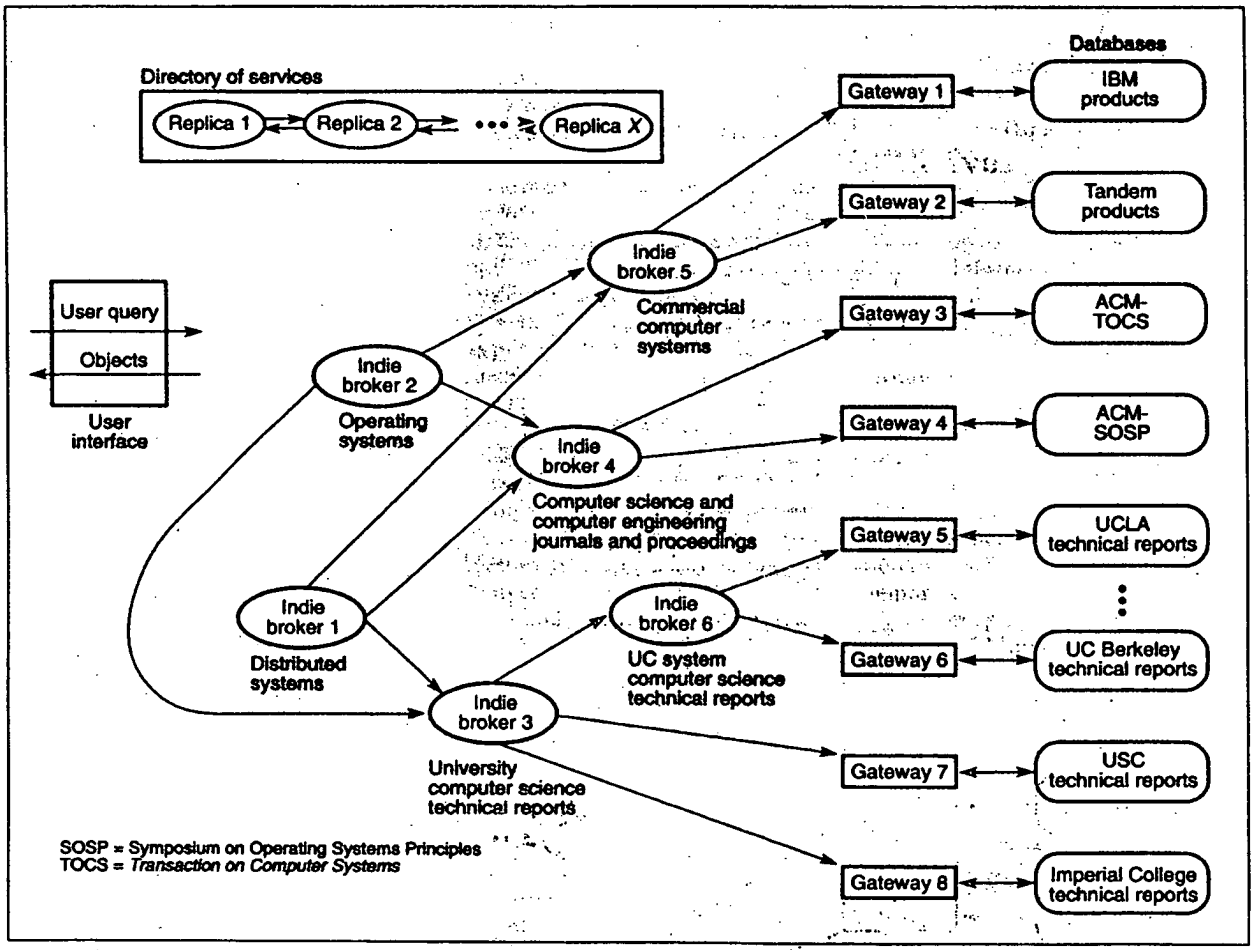

Figure 11. Indie operation.

and replicate itself around the system, and search and manipulate objects. The Knowbot Information System (KIS) is an Internet directory service that understands a number of other directory services, such as X.500, and queries these services on behalf of users.

Alex. This file system developed by Vincent Cate at Carnegie Mellon University provides users with transparent read access to files at anonymous FTP sites. Users can see the collection of Internet anonymous FTP sites and their corresponding directories and files as a hierarchical file system. Intermediate nodes are Internet domains, hosts, or directories within hosts, and leaves are files. Using standard filesystem commands, users can browse through this hierarchy to retrieve files. To obtain reasonable performance, Alex caches information such as machine names, directory information, and the contents of remote files. It also implements a soft consistency mechanism that guarantees that updates include all that occurred in the last 95 percent of the reported age of the file. Alex is currently implemented as an NFS server and already integrates access to Archie. A WAIS database that indexes Readme files scattered throughout the Alex space serves as a substitute for Archie's whatis database. Cate has also built a WAIS database of computer science technical reports available through Alex.

Semantic File Systems. Developed by the Programming Systems Group at the MIT Laboratory for Computer Science, Semantic File Systems integrate associative access into a traditional tree-structured file system. Associative access is achieved by providing file systems with an attribute extraction and query interface. Filters called transducers extract attributes. A transducer takes as input the contents of a file or a directory and produces as output the identifiable objects and their corresponding attributes. An object can correspond to an entire directory, a file, or portions of a file — such as procedures in a source code file — or to individual messages in an e-mail file. Queries consist of Boolean combinations of the desired attribute-value pairs. Transducers and queries produce customized views of the file-system hierarchy called virtual directories that help locate and organize information. A semantic file-system research prototype has been implemented on top of Sun NFS.

Nomenclator. Developed by Joann Ordille and Barton Miller at the University of Wisconsin at Madison, Nomenclator implements attribute-based naming on top of other naming systems. Its access functions are, in essence, servers that periodically traverse the appropriate portions of the underlying name space and other access functions. They build indexes of the objects encountered that satisfy certain properties. The Nomenclator client uses a directory of services called the active catalog to identify access functions pertinent to a user query. Ordille and Miller have built a Nomenclator prototype that uses X.500 as its underlying information repository.

A taxonomy of resource discovery services

We summarize the surveyed tools by presenting a taxonomy of approaches to the resource discovery problem. Schwartz et al. present another taxonomy.[9]

Table 1 lists the features that we used to classify the surveyed discovery tools.

Table 1. A taxonomy for Internet Resource Discovery Services.

Service	Query	Browse	Organize	Granularity	Information Space Organization	Distribution	Directory of Services	Preferred Interfaces
Prospero		✓	✓	Files	DG	Distributed		File system
Gopher		✓		Files	DG	Distributed		Curses-based
WWW		✓	✓	Files and portions of files	DG	Distributed		Hypertext-like
Semantic File System	✓	✓	✓	Portions of files	DG			
X.500	✓		✓	Object descriptors	DG	Distributed		
Alex		✓	✓	Files	DG	Distributed		NFS
Archie	✓			File names	Indexes	Replicated		Xarchie (via Prospero)
WAIS	✓			Objects and object descriptors	Indexes	Distributed	✓	Xwais
Netfind	✓			Information about users	Indexes	Replicated		telnet
Indie	✓		✓	Objects and object descriptors	Indexes	Distributed	✓	Curses-based
Nomenclator	✓			Name server objects	Indexes	Distributed	✓	

Browsing vs. indexing. Services like Gopher and WWW provide users with a browsing interface so that they can navigate the available information space. WWW is also an organizational tool. Like Prospero and traditional file systems, WWW organizes its information space by using links. Users can customize their home cover page to link to interesting information anywhere in the WWW information space. However, users can customize only their starting point in the WWW space, having to follow existing links from then on. Because it is file-system oriented, Prospero is a more flexible organizational tool. It lets users customize their entire information space by using Prospero links and filters. Nevertheless, unlike WWW and hypertext systems in general, in which any node can link to any other node, Prospero can link a directory node only to another directory node or to a file node.

When responding to user queries, services such as Archie, WAIS, Netfind, and Indie search their indexing databases for relevant information. These tools build their indexing databases from information distributed throughout the Internet. Because it is built atop the Distributed Hypertext scheme, Indie also has the potential for allowing users to organize their information space into a distributed hypertext.

Granularity. The granularity with which discovery tools manipulate objects is another distinguishing feature. In Archie, for example, target objects are file names instead of file contents. Therefore, Archie can be used only to locate information stored in files that have meaningful names. WAIS indexes the contents of documents, so users can find interesting documents by submitting keyword-based queries. In Semantic File Systems, users can access self-contained portions of a file, such as procedures in a source code file or individual messages in a mail file.

Organizing the information space. Discovery tools organize searchable data into some kind of information space. Usually, browsing tools organize their information space as a directed graph, with nodes connected by links. Prospero, Gopher, WWW, and Alex belong to this category. WWW is a step toward hypertext systems. It allows links between nodes of any kind. On the other hand, indexing services, such as Archie, Netfind, and Indie, tend to organize searchable information into indexing databases, which allows efficient exhaustive searches.

Data distribution. In addition to the subject of how discovery tools organize data, there is also the question of where this data is stored. In services that employ a graph-based organization, data is usually distributed among geographically dispersed servers. Prospero, Gopher, and WWW belong to this group of services. On the other end of the design spectrum, tools like Archie and Netfind build centralized and replicated indexing databases. Indie and WAIS build distributed indexing databases. WAIS servers store both the indexing database and the corresponding data. Indie's indexing databases are distributed among Indie brokers according to their topic of specialization.

Directories of services. Indie, WAIS, and Nomenclator have implemented directories of services. Discovery sessions can start with a query to the directory of services, which provides users with hints on places to search. The WAIS directory of services knows about all participating WAIS servers and, when responding to a user's query, provides a list of relevant servers. Similarly, Indie's directory responds to user queries with a list of relevant Indie brokers. For scalability and availability purposes, Indie's directory implements Indie's replication mechanism.

The accompanying sidebar shows how to access Internet resource discovery services or where to find their software packages. Figure 12 shows how to get started with resource discovery by using Archie to find out about Gopher distributions.

Directory for the directory

The following information lets users access Internet resource discovery services or find their software packages.

Alex. Service available via Sun-NFS "mount -o timeo=30, retrans=300, soft, intr alex.sp.cs.cmu.edu:/ /alex".

Archie. Service available via Telnet to host Archie.mcgill.ca; log in as Archie.

Gopher. Client and server software available via anonymous FTP from host boombox.micro.umn.edu under directory pub/gopher.

Indie. Software available via anonymous FTP from host jerico.usc.edu under directory pub/indie/indie.tar.1.2.Z.

KIS. Service available via Telnet to host nri.reston.va.us on port 185.

Netfind. Service available via Telnet to host bruno.cs.colorado.edu; login as netfind.

Prospero. Client and server software available via anonymous FTP from host prospero.isi.edu under directory pub/prospero.

WAIS (Wide Area Information Servers). Client and server software available via anonymous FTP from host think.com under directory wais.

WWW (World-Wide Web). Client and server software available via anonymous FTP from host info.cern.ch under directory pub/WWW.

Internet resource discovery services have proliferated because of the continually growing number of hosts on the Internet and the corresponding increase in the amount of available information. Discovery tools can be classified as browsing or indexing tools. Browsing tools organize their information space as a directed graph and

allow users to find data of interest while navigating the information space. Gopher, Prospero, and WWW fall in this category. Indexing services, such as Archie, Netfind, and Indie, organize searchable information into indexing databases and respond to user queries by searching their databases for relevant information.

The first Internet discovery tools were developed independently. However, the current trend is interoperability, whereby users of one discovery service can access information available through other services. For instance, Gopher cli-

1. Contact an Archie server via Telnet as shown in the sidebar "Directory for directories." Note that in the Archie section we showed a sample session using the Xarchie interface. Some of the Archie servers in the US are Archie.rutgers.edu, Archie.unl.edu, and Archie.ans.net.

```
% telnet Archie.rutgers.edu

Trying 128.6.18.15 . . .
Connected to dorm.Rutgers.EDU

login: Archie

ARCHIE: Rutgers University Archive Server [November 20 1992]

Archie>
```

2. Query Archie for Gopher distributions.

```
Archie> prog gopher
```

3. Among other hits, Archie provides the following information.

```
Host boombox.micro.umn.edu (134.84.132.2)
Last updated 00:15 23 Dec 1992

  Location: /pub/slipdial
    DIRECTORY  rwxrwxr-x    512  Nov 23 13:36  gopher
  Location: /pub/pc
    DIRECTORY  rwxr-xr-x    512  Oct 29 01:21  gopher
  Location: /pub
    DIRECTORY  rwxr-xr-x    512  Nov 18 22:27  gopher
```

4. To retrieve the distribution, use anonymous FTP. Note that software distributions are commonly named as <tool name> <distribution number>.tar.Z.

```
% ftp boombox.micro.umn.edu
Connected to boombox.micro.umn.edu.
220 boombox FTP server (Version 4.1 Tue Apr 10 05:15:32 PDT 1990) ready.
Name (boombox.micro.umn.edu:kobraczk): anonymous
331 Guest login ok, send ident as password.
Password: <enter your user id as the password>

230 Guest login ok, access restrictions apply.
ftp> cd pub/gopher/Unix
ftp> get gopher1.1.tar.Z
ftp> quit
```

5. To install Gopher, proceed as follows.

- Create a directory where you want to install gopher `% mkdir gopher`
- Move the distribution to the newly created directory, uncompress it, and restore its directory structure using the tar command.

```
% mv gopher1.1.tar.Z gopher/.
% cd gopher
% uncompress gopher1.1.tar.Z
% tar -xf gopher1.1.tar
```

- Go to the client subdirectory.

```
% cd client verbatim
```

- Make the appropriate environment changes. Essentially, you should edit the Makefile and choose the appropriate machine definition. For instance, if you are installing Gopher on a Sun, you choose uncomment MACHDEFS = -DIS_A_SUN.
- Make, install, and run Gopher.

```
% make
% make install
% gopher
```

Figure 12. How to get started with resource discovery.

ents can retrieve objects from Archie, WAIS, and FTP servers. WWW clients understand FTP archives, Gopher servers, and Archie. Efforts to enhance resource discovery interoperability include the standardization of object identifiers.

Besides enhanced interoperability, future directions in the resource discovery arena include answering questions that range from research-oriented problems — such as how to index information using nontextual indexes, for example, indexing pictures by description and shape — to commercially oriented questions, such as how to bill for data access. ■

References

1. B. Kahle and A. Medlar, "An Information System for Corporate Users: Wide Area Information Servers," *Connexions — The Interoperability Report*, Vol. 5, No. 11, Nov. 1991, pp. 2-9.
2. A. Emtage and P. Deutsch, "Archie: An Electronic Directory Service for the Internet," *Proc. Winter 1992 Usenix Conf.*, Usenix, Sunset Beach, Calif., 1992, pp. 93-110.
3. B.C. Neuman, "The Prospero File System: A Global File System Based on the Virtual System Model," *Computing Systems*, Vol. 5, No. 4, Fall 1992, pp. 407-432.
4. M. McCahill, "The Internet Gopher Protocol: A Distributed Server Information System," *Connexions — The Interoperability Report*, Vol. 6, No. 7, July 1992, pp. 10-14.
5. T. Berners-Lee et al., "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications, and Policy*, Vol. 1, No. 2, Spring 1992, pp. 52-58.
6. M.F. Schwartz, "Experience with a Semantically Cognizant Internet White Pages Directory Tool," *J. Internetworking Research and Experience*, Vol. 1, No. 2, Dec. 1990, pp. 23-50.
7. B. Smetaniuk, "Distributed Operation of the X.500 Directory," *Computer Networks and ISDN Systems*, Vol. 21, 1991, pp. 17-40.
8. P.B. Danzig, S.-H. Li, and K. Obraczka, "Distributed Indexing of Autonomous Internet Services," *Computing Systems*, Vol. 5, No. 4, Fall 1992. Also available via anonymous FTP from caldera.usc.edu: pub/Indie/jcs.ps.Z.
9. M.F. Schwartz et al., "A Comparison of Internet Resource Discovery Approaches," *Computing Systems*, Vol. 5, No. 4, Fall 1992, pp. 461-493.
10. V. Cate, "Alex — A Global File System," *Proc. Usenix File System Workshop*, Usenix, Sunset Beach, Calif., May 1992, pp. 1-11.
11. M.A. Sheldon et al., "Semantic File Systems," *Proc. 13th ACM Symp. Operating Systems Principles*, ACM, New York, 1991, pp. 16-25.
12. J.J. Ordille and B.P. Miller, "Nomenclator Descriptive Query Optimization for Large X.500 Environments," *Proc. ACM SIGComm 91*, ACM, New York, Sept. 1991, pp. 185-196.

Katia Obraczka is a PhD candidate in the Computer Science Department at the University of Southern California in Los Angeles. Her research interests are in computer networks and distributed systems.

Obraczka received a BS in electrical engineering and an MS in computer engineering from the Federal University of Rio de Janeiro, Brazil, in 1981 and 1987, respectively. She received an MS in computer science from the University of Southern California in 1990. She is a member of the IEEE.

Peter B. Danzig is an assistant professor of computer science at the University of Southern California. His current research addresses the measurement and performance debugging of Internet services, distributed system architectures for resource discovery, and mathematical modeling of communication networks.

Danzig received a PhD in computer science from the University of California, Berkeley, in 1990. He holds USC's Innovative Teaching Award and is a member of the program committee for ACM's SIGMetric '93 conference.

Shih-Hao Li is a PhD student in the Computer Science Department at the University of Southern California. He previously worked as a software engineer in the Computer and Communication Laboratory at the Electronics Research and Service Organization, Hsin-chu, Taiwan. His research interests include computer networks, distributed systems, and distributed database systems.

Li received a BS in communication engineering from the National Chiao Tung University, Hsin-chu, Taiwan, and an MS in computer engineering from the University of Southern California. He is a member of the ACM and the IEEE Computer Society.

Katia Obraczka may be contacted at the Computer Science Department, University of Southern California, Los Angeles, CA 90089-0781; e-mail kobraczk@usc.edu.